Neural Computing and Applications
https://doi.org/10.1007/s00521-019-04130-y

ORIGINAL ARTICLE

Multilayer perceptron for short-term load forecasting: from global to local approach

Grzegorz Dudek[1]

Received: 14 February 2018 / Accepted: 4 March 2019

Abstract
Many forecasting models are built on neural networks. The key issues in these models, which strongly translate into the accuracy of forecasts, are data representation and the decomposition of the forecasting problem. In this work, we consider both of these problems using short-term electricity load demand forecasting as an example. A load time series expresses both the trend and multiple seasonal cycles. To deal with multi-seasonality, we consider four methods of the problem decomposition. Depending on the decomposition degree, the problem is split into local subproblems which are modeled using neural networks. We move from the global model, which is competent for all forecasting tasks, through the local models competent for the subproblems, to the models built individually for each forecasting task. Additionally, we consider different ways of the input data encoding and analyze the impact of the data representation on the results. The forecasting models are examined on the real power system data from four European countries. Results indicate that the local approaches can significantly improve the accuracy of load forecasting, compared to the global approach. A greater degree of decomposition leads to the greater reduction in forecast errors.

Keywords Data representation · Forecasting problem decomposition · Neural networks · Short-term load forecasting

1 Introduction

Short-term load forecasting (STLF) aims to predict the future load demand ranging from an hour to a week ahead. This is essential for power system control and scheduling. From an energy generation point of view, STLF is necessary for electric utility operations such as unit commitment, generation dispatch, hydro scheduling, hydrothermal coordination, spinning reserve allocation, interchange and low flow evaluation, fuel allocation and network diagnosis. Since electricity load demand is the basic driver of electricity prices, load forecasting plays an important role in competitive energy markets. Forecast accuracy is a key factor in determining the financial performance of energy companies and other market participants and financial institutions operating in energy markets. Improving the STLF accuracy can significantly reduce the power system operating cost.

Neural networks (NNs) have been widely used in STLF since the early 90 s. This is due to the capacity of NNs to capture the nonlinear relationship between explanatory variables and load. Modeling this relationship is not an easy task because it is unstable in time and strongly dependent on the period of the year, day of the week, hour of the day as well as other factors. NNs have many attractive features, which are extremely useful in hard forecasting problems. These include universal approximation property, capability of learning from examples, several learning paradigms, many architectures, massive parallelism, robustness in the presence of noise and fault tolerance.

Many types of NNs have been used for STLF, such as multilayer perceptron (MLP), radial basis function NN, self-organizing map (SOM), recurrent NNs, generalized regression NNs, probabilistic NNs, adaptive resonance theory NNs, functional link NNs and counterpropagation NNs. A survey of NN applications in STLF can be found in both [1, 2] and comparison of their performances in [3].

✉ Grzegorz Dudek
dudek@el.pcz.czest.pl

1 Department of Electrical Engineering, Czestochowa University of Technology, Al. Armii Krajowej 17, 42-200 Czestochowa, Poland

New solutions in the field of NNs are quickly applied in STLF. For example:

- deep convolutional NN—in [4], a hybrid method based on a deep convolutional NN is introduced for short-term PV power forecasting,
- long short-term memory recurrent NN—in [5], this framework, which is the latest and one of the most popular techniques of deep learning, is proposed for STLF for individual residential households,
- stacked denoising autoencoders—in [6] this model, a class of deep neural networks and its extended version are utilized to forecast the hourly electricity price,
- pooling-based deep recurrent NN—in [7], this network batches a group of customers' load profiles into a pool of inputs and is applied for household load forecasting,
- randomized algorithms for NN training—in [8], a novel hybrid method of probabilistic electricity load forecasting is proposed, including randomized training an improved wavelet NN, wavelet preprocessing and bootstrapping,
- second-order gray NN—in [9], a method based on wavelet decomposition and a second-order gray NN combined with an augmented Dickey–Fuller test is proposed to improve the accuracy of load forecasting,
- echo state networks—in [10], the wavelet echo state network is applied to both STLF and short-term temperature forecasting,
- spiking NN—in [11], spiking NN short-term load forecaster is proposed.

STLF problems are usually decomposed to make them simpler to solve. Subproblems can be modeled using simpler NN architectures, which are easier to learn and generalize better than one big NN designed for the complex original problem. Different methods of decomposition have been proposed. In the early works on neural models for STLF, the separate models for different day types were proposed. A typical example is [12], where seven NNs were built corresponding to the 7 day types from Monday to Sunday. In [13], the load patterns were classified into weekday patterns and weekend-day patterns. The weights of NN were estimated using previous load data for each pattern. Then, the daily period was divided into three parts: 1–9, 10–19 and 20–24, and each part was modeled using individual NN. A similar approach in [14] was applied. For the forecasts of weekdays from Tuesday to Friday, there was one NN for each of these days. For Saturdays, Sundays, Mondays and days after a holiday, the daily period was divided into several parts, which were modeled by separate NNs. Decomposition into day types was also used in [15]. Separate NNs were formed for each day type, and different input variables were used. More recent examples of STLF problem decomposition into day types can be found in [16, 17, 18].

Another common practice is to decompose the load time series into 24 series, each one corresponding to an hour of the day. Then, 24 NNs with a single output learn on these series to provide load forecasts for the next day [19, 20]. In [21] for each day and each hour of the 24 h, a new model was trained on 20 previous days. Thus, this local approach produced models which are competent only for just 1 hour of the current forecasted day. For 24-h-ahead STLF in [10], 24 individual models were built based on wavelet echo state networks, each one for a specific hour of the day.

Forecasting problem decomposition into 12 months was used in [22]. The forecasting model consists of twelve NNs. Each of them performs the final 24-h-ahead load forecast for 1 month of the year. This decomposition is justified due to the differences in the weather during the year. A separate model is also designed for each month in [23]. The proposed models are composed of two hybrid NNs derived from fuzzy NNs.

Several levels of the STLF problem decomposition were used in the forecasting system known as ANNSTLF [24], which by the year 1997 was being used by over 30 utilities in the USA and Canada. This system has three MLP modules: an hourly module with 24 MLPs, a daily module with 7 MLPs and a weekly module. The final load forecast for each hour is found by a proper adaptive combination (based on the recursive least squares) of the forecasts given for this hour by the three MLP modules.

Another approach to STLF problem decomposition relies on clustering the NN inputs and designing a separate NN for each cluster. This approach combines unsupervised and supervised learning concepts. In [25], unsupervised learning is used to identify days with similar daily load and temperature patterns. The training patterns for clustering are selected among patterns from the same period of the year as the forecast day and representing the same day type. For each cluster, a NN with 24 output nodes, which produce 24 hourly loads for the forecast day, is trained. Thus, the relationship between input and output variables is modeled locally within each cluster. In [26], two-stage clustering was used. The SOM identifies similarities among the different load patterns and forms clusters. Then, the k-means algorithm identifies similarities among different clusters and groups similar clusters together. A dedicated MLP for each cluster is trained to properly forecast the load curve. Clustering is also applied on the training set in [10]. After the clusters are formulated, NNs are trained separately with the data of the corresponding clusters.

A completely different approach to the STLF decomposition problem is based on the multilevel wavelet transform (WT). A load time series consisting of both global smooth trends and sharp local variations is decomposed into generalization and details, i.e., low- and high-frequency components. These components are modeled separately by NNs. In [10], each NN gets as inputs a frequency component determined for the day *d*, next-day temperature and day-type index. The output of NN is the predicted next-day value of the corresponding component. NN in the next layer is used to reconstruct the next-day load forecast based on outputs (frequency components) of the NNs from the previous layer. In [27], a forecasting model composed of wavelet transform, NN and an evolutionary algorithm was proposed. Each component is predicted by a combination of NN and the evolutionary algorithm, and then by inverse WT, the hourly load forecast is obtained. The key idea of the STLF method proposed in [28] is to select the similar day's load as the input load, apply WT to decompose it into low-frequency and high-frequency components and then use separate NNs to predict the two components of tomorrow's load.

Among other approaches to the STLF decomposition problem, the following should be mentioned:

- decomposition of the load time series into two parts: the daily average load and the intraday load variation, and modeling these two parts independently using NNs [29],
- decomposition of the STLF problem at the distribution level into "regular" and "irregular" nodes based on load pattern similarities [30]. These node types are then forecasted separately. Each irregular node is forecasted by an individual NN model.
- decomposition into geographical regions due to distinct climate characteristics and consumer bases. In [31], the load forecasting in the four regions of Taiwan is independently performed using NNs.

In this paper, we apply MLP for STLF. This type of network is the most widely used among NNs for modeling nonlinear relationships due its valuable features such as universal approximation property and flexible fitting to the target function. This flexibility is achieved easily by adding new hidden neurons. Training algorithms for MLP have been developed for many years and are still being improved. They can be implemented in many programming languages and environments such as: MATLAB, Python, R, C++, Java and Scala. There are also known effective methods of dealing with overfitting in MLP such as regularization. In the simulation study in this work, we use a powerful and robust Levenberg–Marquardt learning algorithm which interpolates between the Gauss–Newton algorithm and the method of gradient descent. To avoid overfitting, this algorithm is enriched by Bayesian regularization, which provides effective and robust MPL training.

MLP performs very well in STLF against other NNs. In [3], a comparison of several neural STLF models on four datasets is presented. The compared neural models are: MLP, radial basis function NN, generalized regression NN (GRNN), two models based on fuzzy counterpropagation NNs and three models based on self-organizing maps. MLP was trained using the local learning procedure. The forecasting results demonstrate that MLP and GRNN achieve similar level of accuracy for all datasets (with a slight advantage for GRNN). The MLP model performs very well compared to other state-of-the-art models as well as the classical statistical models: ARIMA and exponential smoothing (see comparison in Table 9 in Sect. 4).

In this work, we analyze and compare the global and local approaches to building the STLF models based on MLP. In the global approach, we built a model competent for each hour of the day and each day of the week. Local approaches include four methods of the forecasting problem decomposition:

- forecasting for each day of the week separately using seven NNs,
- forecasting for each hour of the day separately using 24 NNs,
- forecasting for each day of the week and each hour of the day separately using 168 NNs,
- forecasting separately for each forecasting task using NN built only for this task.

The daily curves of the load in the proposed models are preprocessed to filter out the trend, weekly and annual cycles, and are introduced to the models as input and output variables. Depending on the approach, other input data are used: period of the year, day of the week and hour of the day. The performance of the model is dependent on how data are represented. For day of the week, six methods of coding are considered, while for hour of the day, five methods are considered. In the experimental part of this work, global and local approaches are examined as well as data representation methods.

The paper is organized as follows. In Sect. 2, data representation ways are described. The forecasting models in global and local versions are presented in Sect. 3. In Sect. 4, the models using different data representation ways are tested on real load data and compared. Finally, Sect. 5 concludes this paper.

2 Data representation

The proposed neural models generate forecast of the load at timepoint t of the day $i+\tau$, where i is the number of the current day and τ is the forecast horizon in days. The forecasted day is located in some period of the year and represents some day of the week. It is assumed that the load pattern of the current day i is available and can be used as an input of the model. It is represented by a vector $\mathbf{x} = [x_1\ x_2\ \ldots\ x_n]$, where n is a number of timepoints in a daily period (24 for hourly resolution). The load pattern of the forecasted day $i+\tau$ is represented by a vector $\mathbf{y} = [y_1\ y_2\ \ldots\ y_n]$. A neural model (MLP) generates one of the **y**-vector component as an output: y_t. The timepoint number t, hour of the day in our case, is represented by a vector **h**. The forecasted day is located in the period of the year represented by a vector **p**, and its type (Monday, …, Sunday) is represented by a vector **d**. Vectors **x**, **p**, **d** and **h** can be used as inputs of NNs, and component y_t is an output. The methods of representation of input and output data are shown below.

2.1 Representation of load time series

The i-th daily period of the load time series $\mathbf{L}_i = [L_{i,1}\ L_{i,2}\ \ldots\ L_{i,n}]$ is represented by the input vector $\mathbf{x}_i = [x_{i,1}\ x_{i,2}\ \ldots\ x_{i,n}] \in X = \mathbb{R}^n$, which is an normalized version of the load vector $\mathbf{L}_i$. The components of input vector are defined as follows [32]:

$$x_{i,t} = \frac{L_{i,t} - \bar{L}_i}{D_i} \tag{1}$$

where: $i = 1, 2, \ldots, N$ is the daily period number, $t = 1, 2, \ldots, n$ is the time series element number in the period i, $L_{i,t}$ is the t-th load time series element in the period i, $\bar{L}_i$ is the mean load in the period i and $D_i = \sqrt{\sum_{l=1}^{n} (L_{i,l} - \bar{L}_i)^2}$ is the dispersion of the time series elements in the period i.

Note that after normalization, all load vectors $\mathbf{L}_i$ have unity length, zero mean and the same variance. Thus, the load time series, which is nonstationary in mean and variance, is represented by **x**-vectors having the same mean and variance. They carry information about the shape of the daily load curve. (The trend and weekly and annual variations are filtered.)

The forecasted daily period $i+\tau$: $\mathbf{L}_{i+\tau} = [L_{i+\tau,1}\ L_{i+\tau,2}\ \ldots\ L_{i+\tau,n}]$ is represented by an output vector $\mathbf{y}_i = [y_{i,1}\ y_{i,2}\ \ldots\ y_{i,n}] \in Y = \mathbb{R}^n$. The **y**-vector components are defined as follows:

$$y_{i,t} = \frac{L_{i+\tau,t} - \bar{L}_i}{D_i} \tag{2}$$

where: $i = 1, 2, \ldots, N$, $t = 1, 2, \ldots, n$.

In (2), the coding variables $\bar{L}_i$ and D_i determined for the day i are used instead of the day $i+\tau$. This is because their values for the day $i+\tau$ are unknown at the moment of forecasting. Using their known values for the i-th day enables us to calculate the forecasted load value. To do so, transformed Eq. (2) is used:

$$\widehat{L}_{i+\tau,t} = \widehat{y}_{i,t} D_i + \bar{L}_i \tag{3}$$

where $\widehat{y}_{i,t}$ is the t-th component of the **y**-vector forecasted by the NN.

As the daily periods of the load time series are coded as **x**- and **y**-vectors, we unify the input and output data and simplify the relationships between them. This is further discussed in [32]. The expected result is a simpler and more accurate forecasting model.

2.2 Representation of period of the year

A period of the year is represented by two-component vector **p**:

$$\mathbf{p} = \left[\sin\left(2\pi\frac{\#(i+\tau)}{366}\right) \quad \cos\left(2\pi\frac{\#(i+\tau)}{366}\right)\right] \tag{4}$$

where $\#(i+\tau) = 1, 2, \ldots, 366$ is the forecasted day number in the year.

Days which are in the same position in time in a yearly cycle have similar values of their **p**-vector components. This way of period of the year representation was used in many works, see [12, 10, 29].

2.3 Representations for day of the week

Index of the day of the week δ has seven values: 1 for Monday, 2 for Tuesday, … and 7 for Sunday. Six coding ways for δ are considered ($dc = 1, 2, \ldots, 6$). In the first one, the index of the day is scaled to the range [1/7, 1] (see $dc = 1$ in Table 1). In this representation, successive days of the week, from Monday to Sunday, have successive values of the scaled index. The most distant days are Mondays and Sundays for which $\mathbf{d}^1 = 1/7$ and 1, respectively.

In the second representation, sine and cosine functions are used for coding the day index in a similar way as in the case of coding the period of the year (see $dc = 2$ in Table 1). Note that in this "periodic" representation, the

Table 1 Representations for day of the week

dc	1	2	3	4	5	6
	$\mathbf{d}^1$	$\mathbf{d}^2$	$\mathbf{d}^3$	$\mathbf{d}^4$	$\mathbf{d}^5$	$\mathbf{d}^6$
Monday	$\delta/7$	$[\sin(2\pi\delta/7)$	1000000	001	001	00
Tuesday		$\cos(2\pi\delta/7)]$	0100000	010	011	01
Wednesday			0010000	011	010	01
Thursday			0001000	100	110	01
Friday			0000100	101	111	01
Saturday			0000010	110	101	10
Sunday			0000001	111	100	11

last day of the week has similar **d**-vector values as the first day of the week.

The third representation uses seven bits. The δ-th day of the week is represented by a vector having one at the δ-th position and zeros at remaining positions (see dc = 3 in Table 1).

In the fourth representation, index δ is encoded using the natural binary system. Three bits are needed for coding seven values of the index (see dc = 4 in Table 1). Note that in this representation, two neighboring values of the index can differ significantly in the binary space. For example, values 3 and 4 are represented by vectors [011] and [100], respectively, where all three bits are different. This disorder of regularity can affect learning of the neural network. To improve this, in the fifth representation, Gray code is used, in which two adjacent values of the index differ in only one bit in the binary space (see dc = 5 in Table 1).

In the last representation, days of the week are grouped according to the load pattern similarity. Four groups are assumed: (1) Mondays, (2) Tuesdays–Fridays, (3) Saturdays and (4) Sundays. The group number is binary encoded (see dc = 6 in Table 1).

2.4 Representations for hour of the day

Hour of the day, t = 1, 2, …, 24, is represented in five ways (hc = 1, 2, …, 5) corresponding to the first five ways of the day of the week representation, see Table 2. In the third representation, 24 bits are needed. The fourth and fifth representations, i.e., natural binary coding and Gray coding, respectively, both use five bits.

The features of the coding method have an influence on NN learning and its ability to map inputs into outputs. The first important feature of coding is the adjacency property: adjacent values in the original space are represented by adjacent values in the code space. All representation methods have this property except for natural binary coding. The second feature is the periodicity property: the beginning and ending values in the original space are represented by similar values in the code space. This property can be important for periodically changing variable such as load in daily, weekly and annual periods. This property is evident only for representations based on sine and cosine functions. Another important feature is the number of components of the code vector. This corresponds to the number of free parameters of NN (weights connecting inputs with hidden neurons), which are used to map the encoded variable into the output variable (in the context of other input variables of course). The more components (inputs) for a variable, the more weights for it, which enables the network to model more complex relationships. In the case of day of the week and hour of the day, the most inputs are delivered by the third representation methods: 7 for $\mathbf{d}^3$ and 24 for $\mathbf{h}^3$, respectively. In the first representation method of these variables ($\mathbf{d}^1$ and $\mathbf{h}^1$, respectively), there is only one component. One component is also used in the case of the load time series representation. (Each load value is represented by one component of **x**-vector.)

In Sect. 4, the representation methods are tested experimentally.

Table 2 Representations for hour of the day

hc	1 $\mathbf{h}^1$	2 $\mathbf{h}^2$	3 $\mathbf{h}^3$	4 $\mathbf{h}^4$	5 $\mathbf{h}^5$
Hour 1	$t/24$	$[\sin(2\pi t/24)$	1000…0	00001	00001
Hour 2		$\cos(2\pi t/24)]$	0100…0	00010	00011
Hour 3			0010…0	00011	00010
…			…	…	…
Hour 24			0000…1	11000	10100

3 Forecasting models for STLF based on MLPs

Five variants of the forecasting models based on MLP are examined, v.1–v.5. They correspond to the method of the STLF problem decomposition. In each case, the forecasting task is to forecast the power system load at hour χ = 1, 2,…, 24 of the day $i + \tau$.

For all cases, MLP with one hidden layer is used. The Levenberg–Marquardt algorithm with Bayesian regularization is applied for learning MLPs. In this algorithm, combination of squared errors and net weights is minimized. This expansion of the cost function to search not only for the minimal error, but also for the minimal error using the minimal weights prevents overfitting. Compared to other methods of improving generalization in NNs, Bayesian regularization gives very good results [33].

3.1 Global model (v.1)

This model is built for forecasting the system load at each hour of the day, t = 1, 2,…, 24, and for each day of the week, δ = 1, 2,…, 7. Thus, the model is competent for every forecasting task. The model inputs are:

- **x**-vector for the day i, $\mathbf{x}_i$,
- period of the year from which the forecasted day is, $\mathbf{p}_{i+\tau}$,
- type of the forecasted day, $\mathbf{d}_{i+\tau}$, and
- forecasted hour, $\mathbf{h}_t$.

The input pattern is of the form [$\mathbf{x}_i$ $\mathbf{p}_{i+\tau}$ $\mathbf{d}_{i+\tau}$ $\mathbf{h}_t$]. The output is $y_{i,t}$. The model learns on the training set including 24 m samples:

$$\{([\mathbf{x}_i\mathbf{p}_{i+\tau}\mathbf{d}_{i+\tau}\mathbf{h}_t], y_{i,t}) : i = 1, 2, \ldots, m, \quad t = 1, 2, \ldots, 24\} \tag{5}$$

and then is tested on the test set of size 24v samples:

$$\{([\mathbf{x}_i\mathbf{p}_{i+\tau}\mathbf{d}_{i+\tau}\mathbf{h}_t], y_{i,t}) : i = m+1, m+2, \ldots, m+v, \\ t = 1, 2, \ldots, 24\} \tag{6}$$

where m is the number of days in the training set and v is the number of days in the test set.

In the experimental part of the work, the training set contains samples from the first two or three years of the data and the test set contains samples from the next year. The model which is trained only once on the training set is competent for the entire test period, i.e., for $i = m + 1$ to $m + v$, where v is 366 for 1-year test period.

3.2 Separate NN for each day of the week (v.2)

In this approach, for each day of the week, a separate NN is trained. Thus, there is no need to introduce the type of the forecasted day $\mathbf{d}_{i+\tau}$ as input. The input pattern is in the form: [$\mathbf{x}_i$ $\mathbf{p}_{i+\tau}$ $\mathbf{h}_t$]. The NN for the day-type δ learns on the training set:

$$\{([\mathbf{x}_i\mathbf{p}_{i+\tau}\mathbf{h}_t], y_{i,t}) : i \in \Delta^{1\ldots m}, \quad t = 1, 2, \ldots, 24\} \tag{7}$$

and then is tested on the test set:

$$\{([\mathbf{x}_i\mathbf{p}_{i+\tau}\mathbf{h}_t], y_{i,t}) : i \in \Delta^{m+1\ldots m+v}, \quad t = 1, 2, \ldots, 24\} \tag{8}$$

where $\Delta^{a\ldots b} = \{i = a, a + 1, \ldots, b: type(i + {}_\tau) = \delta\}$ and $type(i + {}_\tau)$ is the day of the week index of the forecasted day.

Note that NN for the day-type δ learns on the training patterns representing just this type of the day. Each NN is competent for the entire test period but only for one of the seven days of the week.

3.3 Separate NN for each hour of the day (v.3)

For each hour of the day, a separate NN is built. Inputs do not include the hour of the day $\mathbf{h}_t$. The input pattern is in the form: [$\mathbf{x}_i$ $\mathbf{p}_{i+\tau}$ $\mathbf{d}_{i+\tau}$]. The NN for hour χ learns on the training set:

$$\{([\mathbf{x}_i\mathbf{p}_{i+\tau}\mathbf{d}_{i+\tau}], y_{i,\chi}) : i = 1, 2, \ldots, m\} \tag{9}$$

and then is tested on the test set:

$$\{([\mathbf{x}_i\mathbf{p}_{i+\tau}\mathbf{d}_{i+\tau}], y_{i,\chi}) : i = m+1, m+2, \ldots, m+v\} \tag{10}$$

Thus, 24 NNs are built. Each of them is competent for the entire test period but only for one of the 24 h of the day. Note that input patterns [$\mathbf{x}_i$ $\mathbf{p}_{i+\tau}$ $\mathbf{d}_{i+\tau}$] for all 24 NNs are the same, but target outputs are different. NN for hour χ has χ-th component of $\mathbf{y}$-vector as its output: $y_{i,\chi}$.

3.4 Separate NN for each day of the week and hour of the day (v.4)

In this case, we built 7 × 24 = 168 NNs. Each of them is competent for the entire test period but only for a selected hour of the selected day of the week. The input pattern includes $\mathbf{x}$-vector and $\mathbf{p}$-vector: [$\mathbf{x}_i$ $\mathbf{p}_{i+\tau}$]. The training set is defined as:

$$\{([\mathbf{x}_i\mathbf{p}_{i+\tau}], y_{i,\chi}) : i \in \Delta^{1\ldots m}\}, \tag{11}$$

and the test set is defined as:

$$\{([\mathbf{x}_i\mathbf{p}_{i+\tau}], y_{i,\chi}) : i \in \Delta^{m+1\ldots m+v}\} \tag{12}$$

In this case, NN designed for the day-type δ and hour χ learns on training patterns corresponding to days of type δ and hour χ.

3.5 Separate NN for each forecasting task (v.5)

In this approach, the NN is constructed for each forecasting task. The input pattern is composed only of $\mathbf{x}_i$. The output is $y_{i,\chi}$. For a given forecasting task—the forecast the power system load at hour χ of the day $i + \tau$, whose day type is δ, and number is q—the training set is built in the form:

$$\{(\mathbf{x}_i, y_{i,\chi}) : i \in \Delta^{1\ldots q-1}\}, \tag{13}$$

and one-element test set is:

$$\{(\mathbf{x}_q, y_{q,\chi})\} \tag{14}$$

Note that in the case of this model, the historical period from which the training set is generated is not limited to m days, but also contains recent days from $m + 1$ to $q - 1$, i.e., all available data, up to the day, which is the last day from the available history. Newer information hidden in data, from $i = m + 1$ to $q - 1$, is used for building a forecasting model for the day q. In the case of models described above, this information is not used. For example, when we forecast last day of the 1-year test period (v = 366) using models v.1–v.4, we use information older than 1 year to train the model. Model v.5 learns on the most recent information about the load time series. We expect this to increase model performance.

In [34], a similar neural model was proposed, but it was trained locally on the training samples ($\mathbf{x}_i$, $y_{i,t}$), where $\mathbf{x}$-vectors belong to the set of k nearest neighbors of the current $\mathbf{x}$-vector. During experiments conducted as part of this work, we noticed that increasing k counterintuitively improves results. Thus, in this study, we do not limit the number of samples to k nearest neighbors, but train our model using all samples from history [see $i \in \Delta^{1 \ldots q-1}$ in (13)].

4 Simulation study

In this section, the neural models v.1–v.5 as well as data representation methods are evaluated on real data: four data sets containing hourly loads of Polish (PL), British (GB), French (FR) and German (DE) power systems in the period 2012–2015. (The source of data is www.entsoe.eu.) A one-day-ahead STLF problem is considered ($\tau = 1$). Then, the forecast accuracy of the best variant of the proposed neural model is compared with levels of accuracy achieved by state-of-the-art models applied to STLF. The comparative models include neural networks, linear regression, non-parametric regression, clustering-based models, artificial immune systems, ARIMA, exponential smoothing and the naïve model.

The optimization and training procedures for neural models in variants v.1–v.4 are as follows. First, the number of hidden neurons and the best methods of input data representation are selected. To do this, the training is repeated for each variant of data representation as well as for #neurons = 1, 2,…, g. For example, in the case of model v.1, we have four input variables: $\mathbf{x}_i$, $\mathbf{p}_{i+\tau}$, $\mathbf{d}_{i+\tau}$, $\mathbf{h}_t$. The day of the week is encoded using one of six ways, and the hour of the day is encoded using one of five ways. Thus, there are $5 \times 6 = 30$ combinations of input data representation. For each combination, we train the model composed of 1, 2,…, g neurons. The training set is created using data from the period 2012–2013. After training, the model is tested on data from 2014. The best model is selected having the lowest error on the test period. Then, the best model is trained on data from the period 2012–2014 and then tested on data from 2015. The mean error for the forecasting tasks from the test period (2015) is a measure of the model quality.

In the case of the neural model in variant v.5, the optimization and training procedures are different. This model is trained for each forecasting task from the test period (2015) independently. First, to select the number of hidden neurons, ten models are built for ten forecasting tasks from the history, which are similar to the current forecasting task. The history from which these ten tasks are selected is limited to the period covering year 2014 and the period of 2015 preceding the current forecasting task for the day q. By similar forecasting tasks to the current one, we mean tasks for the same day-type δ as the current task and having $\mathbf{x}$-vectors similar to the $\mathbf{x}$-vector of the current task ($\mathbf{x}_q$). The similarity measure between $\mathbf{x}$-vectors is the Euclidean distance. For example, when the forecasting task concerns July 1, 2015, a Wednesday, we select ten similar forecasting tasks from the period from January 1, 2014, to June 29, 2015. For PL data ,these selected tasks are (ranked in the order of similarity): June 11, 2014, June 25, 2014, June 17, 2015, June 24, 2015, July 2, 2014, July 16, 2014, July 9, 2014, June 18, 2014, July 30, 2014 and June 10, 2015. As we can see, these days are from the same period of the year as the current forecasted day. The model learns for each of the ten similar tasks independently on the training set generated from the historical data according to (13), where now q is the similar day number. The training is repeated for #neurons = 1, 2,…, g, and the optimal number of neurons is selected for which the mean error determined on similar tasks is minimal. Then, the optimal model learns on the training set (13) generated from the period starting on January 1, 2012, and the forecast for the current task is generated. Thus, for each forecasting task from 2015, a separate NN is created and optimized on the historical forecasting tasks which are most similar to the current one.

The error measure applied in this study is the mean absolute percentage error (MAPE), which is traditionally used as an error measure in STLF. Atypical days such as public holidays are excluded from the training and test sets (between 10 and 20 days in a year).

The errors for the model v.1 with different methods of encoding of the day type (dc) and hour of the day (hc) are shown in Fig. 1. Note that in most cases binary coding for the hour of the day provides much better results than other ways of coding. In the case of PL and FR data, we get the best results when hour of the day is coded on 24 bits and the day of the week is coded using sine and cosine functions. For GB data, best results are obtained when the hour of the day is coded using natural binary code and the day type is coded on seven bits. The best results for DE data are achieved when the hour of the day is represented in natural binary code, and the day type is encoded in two bits representing one of four groups ($dc = 6$). The best methods of data representation are summarized in Table 3, which also shows the optimal numbers of hidden neurons and errors for 2014 and 2015. The error for 2015, as an error on data unseen during the learning and optimization procedures, is the right measure of the model's performance.

Table 4 shows results for the model in variant v.2. Note that in optimization procedures in all cases, binary coding was selected for the hour of the day. The forecasts for Mondays are usually less accurate than forecasts for other days of the week. Unexpectedly, mean errors generated by model v.2 are higher than errors generated by model v.1 for all datasets. This leads to the conclusion that the decomposition of the forecasting problem on separate models for each day of the week is not a good idea.

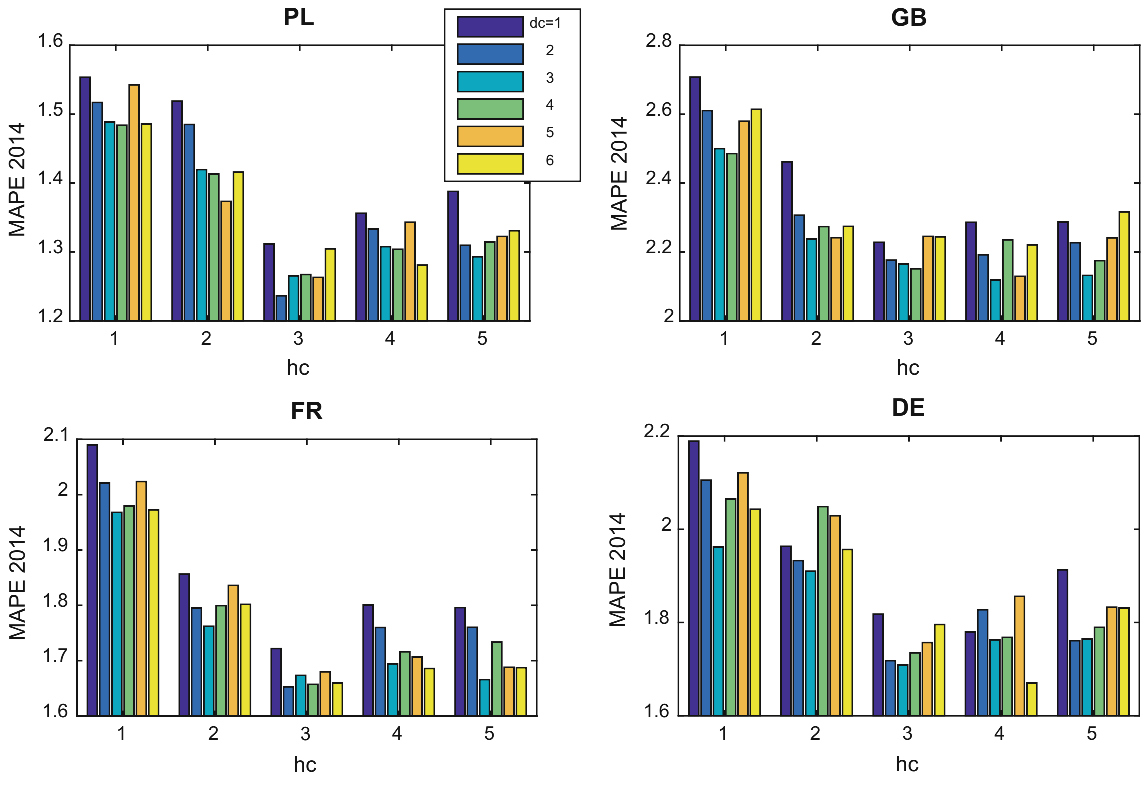

Fig. 1 Model v.1 performance with different methods of day of the week and hour of the day encoding

Table 3 Results for variant 1

	PL	GB	FR	DE
dc	2	3	2	6
hc	3	4	3	4
#neurons	12	13	8	14
MAPE 2014	1.24	2.12	1.65	1.67
MAPE 2015	1.35	2.81	1.70	1.66

Table 4 Results for variant 2

	Mon	Tue	Wed	Thu	Fri	Sat	Sun	Mean
PL								
hc	4	5	5	3	4	4	4	
#neurons	6	13	10	4	16	10	9	9.71
MAPE 2014	2.07	1.13	1.2	1.12	1.24	1.29	1.28	1.33
MAPE 2015	2.15	1.11	1.24	1.28	1.38	1.49	1.49	1.45
GB								
hc	3	4	4	4	4	5	5	
#neurons	5	9	8	9	7	6	6	7.14
MAPE 2014	2.38	3.12	2.56	3.05	2.66	3.36	3.34	2.92
MAPE 2015	3.88	3.53	3.49	3.83	3.23	3.41	3.43	3.54
FR								
hc	4	5	4	5	5	4	4	
#neurons	5	8	9	7	8	6	7	7.14
MAPE 2014	2.36	1.76	1.72	1.96	1.56	1.85	1.77	1.85
MAPE 2015	2.32	1.93	1.78	2.07	1.97	1.88	1.68	1.95
DE								
hc	3	3	4	4	4	5	4	
#neurons	1	6	7	9	10	8	6	6.71
MAPE 2014	2.64	1.2	1.39	1.39	1.59	1.56	1.56	1.62
MAPE 2015	2.27	1.86	2.33	1.58	1.84	1.88	1.64	1.91

Results for the models in variants v.3 and v.4 are presented in Tables 5 and 6, respectively. Note that the optimal number of neurons for v.4 is about 2 or 3 on average and for v.3 about 5–7. This is because models v.4 are "more local" than v.3, and the relationships between input and output variables are easier and can be modeled by simpler NN. The global approach v.1 needs 8–14 neurons, and variant v.2, which is decomposed on 7 NNs needs 6–10 neurons on average. Thus, we can expect that the "most local" approach v.5 will need even fewer neurons than v.4. In Table 7, errors are shown for models v.5 in three variants:

(a) optimization procedure is carried out to select the optimal number of neurons for each forecasting task (#neurons = 1, 2,…, 5),

Table 5 Results for variant 3

	Hour	1	2	3	4	5	6	7	8	9	10	11	12
PL	*dc*	3	6	4	5	4	6	4	3	6	6	5	3
	#neurons	2	7	11	8	10	7	6	3	8	5	8	7
	MAPE 2014	0.56	0.66	0.71	0.74	0.84	0.92	1.09	1.21	1.23	1.16	1.23	1.25
	MAPE 2015	0.49	0.67	0.74	0.79	0.92	1.00	1.19	1.21	1.60	1.27	1.33	1.33
GB	*dc*	2	5	1	6	6	5	5	6	4	6	6	3
	#neurons	4	3	6	2	5	5	8	5	7	8	3	5
	MAPE 2014	0.43	0.94	1.10	1.38	1.63	1.82	1.99	1.90	1.96	2.09	2.69	3.36
	MAPE 2015	0.66	0.99	1.31	1.54	1.63	2.05	2.18	2.25	2.35	2.37	2.69	3.78
FR	*dc*	1	1	4	2	2	1	4	3	3	2	3	2
	#neurons	6	14	7	10	18	16	8	4	4	10	7	9
	MAPE 2014	0.38	0.62	0.78	0.86	0.96	1.15	1.39	1.54	1.41	1.33	1.36	1.47
	MAPE 2015	0.41	0.60	0.85	1.04	1.10	1.30	1.54	1.59	1.42	1.59	1.44	1.59
DE	*dc*	6	4	3	6	6	6	3	6	3	5	6	4
	#neurons	1	10	3	5	15	3	7	10	5	6	8	5
	MAPE 2014	0.41	0.58	0.69	0.80	0.84	0.94	1.25	1.29	1.31	1.23	1.30	1.29
	MAPE 2015	0.38	0.65	0.73	0.82	0.92	0.98	1.39	1.49	1.55	1.32	1.41	1.48

	Hour	13	14	15	16	17	18	19	20	21	22	23	24	Mean
PL	*dc*	4	6	4	4	4	5	2	1	1	3	3	4	
	#neurons	7	5	6	7	3	7	7	9	8	5	8	6	6.67
	MAPE 2014	1.31	1.38	1.38	1.50	1.48	1.40	1.33	1.30	1.17	1.08	1.12	1.14	1.13
	MAPE 2015	1.44	1.50	1.44	1.64	1.65	1.43	1.36	1.37	1.36	1.19	1.19	1.22	1.22
GB	*dc*	5	6	5	5	5	5	5	3	3	2	4	3	
	#neurons	6	7	5	7	8	4	3	3	2	3	2	3	4.75
	MAPE 2014	3.92	4.24	4.42	4.39	4.09	3.52	2.93	2.67	2.48	2.26	2.12	2.25	2.52
	MAPE 2015	4.21	4.99	5.19	4.72	4.09	3.84	3.11	2.98	2.51	2.33	2.37	2.62	2.78
FR	*dc*	4	5	3	4	2	2	1	2	5	2	3	2	
	#neurons	9	5	9	6	6	5	5	4	3	6	3	6	7.50
	MAPE 2014	1.56	1.80	1.96	2.10	2.21	2.00	1.88	1.76	1.71	1.71	1.64	1.67	1.47
	MAPE 2015	1.67	1.87	2.41	2.22	2.37	2.11	1.93	1.80	1.81	1.83	1.64	1.73	1.58
DE	*dc*	3	3	3	4	3	3	3	1	2	1	3	3	
	#neurons	2	5	5	6	7	6	4	7	12	5	2	2	5.88
	MAPE 2014	1.40	1.48	1.63	1.59	1.61	1.54	1.41	1.32	1.34	1.26	1.31	1.36	1.22
	MAPE 2015	1.40	1.59	1.73	1.73	1.73	1.54	1.44	1.49	1.56	1.26	1.33	1.36	1.30

Table 6 Results for variant 4

	PL	GB	FR	DE
#neurons	3.48	2.97	2.77	2.40
MAPE 2014	1.16	2.64	1.53	1.22
MAPE 2015	1.20	2.73	1.56	1.24

Table 7 MAPE for variant 5

	PL	GB	FR	DE
v.5a	1.17*	2.63*	1.63*	1.25*
v.5b	1.27	2.53*	1.61*	1.26
v.5c	1.16*	2.57*	1.62*	1.27*

(b) there is no optimization phase—there is one neuron in the hidden layer for all forecasting tasks, and
(c) there is no optimization phase—there are two neurons in the hidden layer for all forecasting tasks.

The best results for each dataset in Table 7 are marked with an asterisk (best results were confirmed by Wilcoxon rank sum test with 5% level of significance). As we can see from this table, there is no difference in errors between variants v.5a and v.5c. But note that variant v.5c does not require the time-consuming optimization phase.

A comparison of the proposed approaches v.1–v.5 is given in Table 8. The lowest errors are for PL data, then for DE and FR data. In contrast, errors for GB data are up to two times higher than for the other data sets. As in Table 7, asterisks indicate best results, which have been statistically confirmed. The most accurate approaches are v.5 and v.4. These approaches do not use the day of the week and hour of the day as inputs which means that searching for the best way of representation is unnecessary for these models. In variant v.5c, searching for the optimal number of neurons is unnecessary as well, meaning the model in this variant can be built very fast.

In Fig. 2, errors for each hour of the day and day of the week are shown. In the case of PL, FR and DE data, the most outlying errors are for Mondays. Errors for the earliest hours of the day are the lowest, while for hours 12–18, corresponding to the peak load, they are highest.

Finally, we compare the forecast accuracy of the best variant of the proposed neural model, v.5, with the accuracy of the state-of-the-art models applied to STLF. The models were tested in one-day-ahead STLF problems on the following datasets:

- PL—hourly load time series of the Polish power system in the period of 2002–2004,
- FR—half-hourly load time series of the of the French power system in the period of 2007–2009,
- GB—the half-hourly load time series of the British power system in the period of 2007–2009 and
- VC—the half-hourly load time series of the power system of Victoria, Australia, in the period of 2006–2008.

The comparative models are:

- RBFNN—radial basis function NN,
- GRNN—generalized regression NN,
- FCPNN1—fuzzy counterpropagation NN, instar clustering variant,
- FCPNN2—fuzzy counterpropagation NN, outstar clustering variant,
- SOM1—self-organizing map, concatenated x- and y-patterns clustering variant,
- SOM2—self-organizing map, independent x- and y-patterns clustering variant,
- SOM3—self-organizing map, y-patterns clustering variant,
- PCR—principal components regression,
- PLSR—partial least-squares regression,
- N-WE—Nadaraya-Watson estimator,
- FNM—fuzzy neighborhood model,
- FP1 + k-means—model based on k-means clustering of concatenated x- and y-patterns,
- FP2 + k-means—model based on k-means clustering of x- and y-patterns independently,
- AIS1—artificial immune system working on concatenated x- and y-patterns,
- AIS 2—artificial immune system working on separate populations of x- and y-patterns,
- AISLFS—artificial immune system with local feature selection,
- ARIMA—auto regressive integrated moving average model ARIMA$(p, d, q) \times (P, D, Q)_v$,
- ES—exponential smoothing state space model,
- Naïve—naïve model: the forecasted daily curve is the same as seven days ago.

Table 8 MAPE for the best models in each variant

	v.1	v.2	v.3	v.4	v.5
PL	1.35	1.45	1.22	1.20	1.16*
GB	2.81	3.54	2.78	2.73	2.57*
FR	1.70	1.95	1.58	1.56*	1.62
DE	1.66	1.91	1.30	1.24*	1.27*

The first seven models are based on different types of NNs and were described in detail in [3]. The next two models, PCR and PLSR [35], are linear regression models in which the components of the input patterns are constructed by the linear combination of the original components. In these models, the relationship between input and output patterns is modeled locally in the neighborhood of a query pattern. N-WE and FNM are nonparametric regression models [36], where the regression curve is a linear combination of **y**-vectors weighted by the function which nonlinearly maps the distance between **x**-vectors. FP1 + k-means and FP2 + k-means [36] aggregate the x- and y-patterns into clusters, assign the query pattern to the cluster and reconstruct the forecasted y-pattern from the cluster characteristics. Artificial immune systems, AIS1, AIS2 and AISLFS [36, 37], are biologically inspired computation methods, where the forecasting problem is solved in the immune memory creation process. In these models, antibodies are the recognition and prediction units, which memorize features of the time series and reconstruct the forecasted pattern. For ARIMA and ES, the time series were decomposed for each hour of the day (or half hour, depending on the time series resolution) and a separate series was created. In this way, a daily seasonality was eliminated. ARIMA or ES was used for the independent modeling of these series. In the above list of forecasting models, except ARIMA, ES and Naïve, the time series are

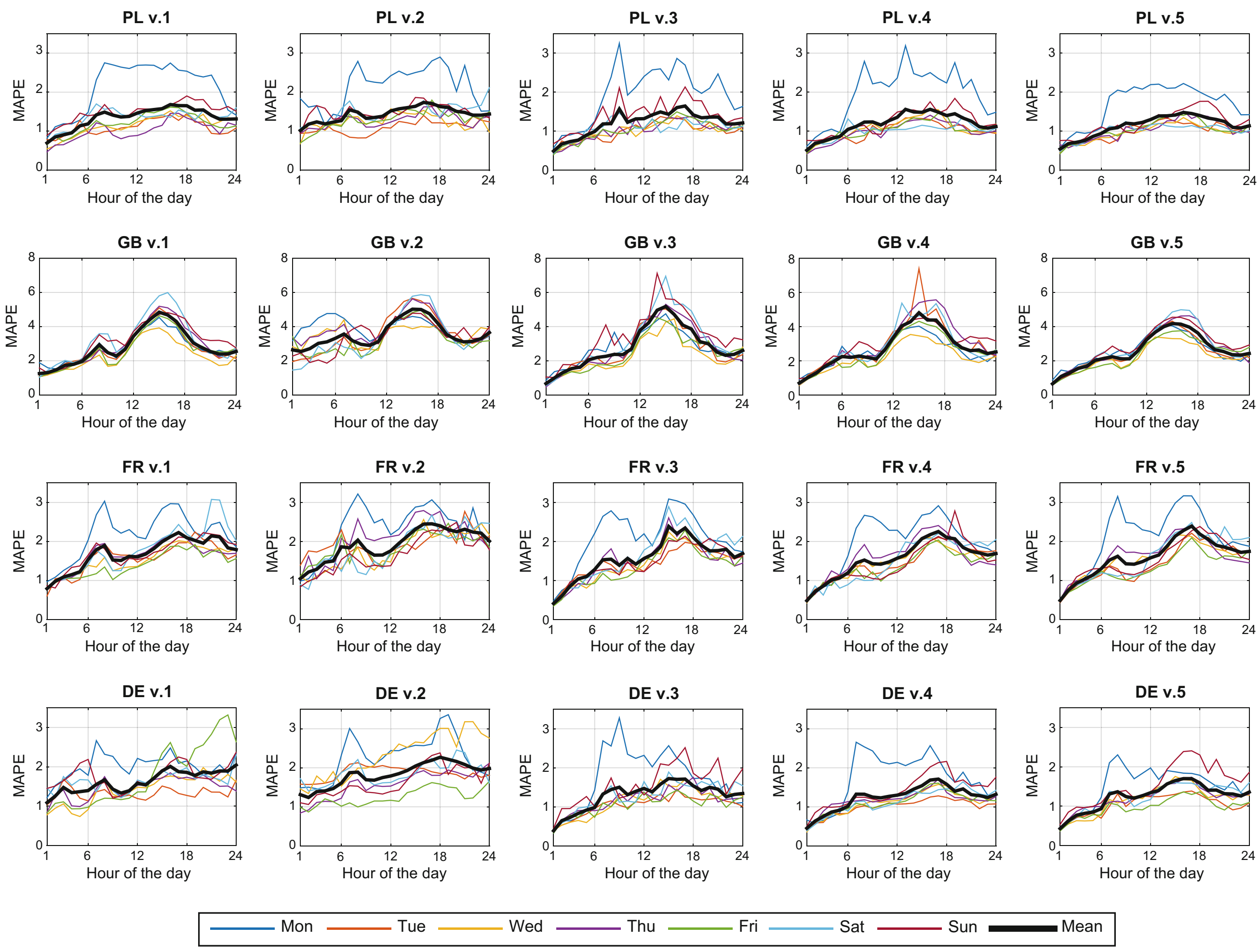

Fig. 2 Forecast errors for different hours of the day and days of the week

represented by patterns defined in the same way as in this work [see Eqs. (1) and (2)].

Table 9 gives the forecast errors for the test period (last year of the time series) for the models listed above. The lowest errors are in bold. As we can see from this table, MLP v.5 is among the most accurate models. It should also be noted that, in most cases, pattern-based methods outperform the classical methods such as ARIMA and ES.

5 Conclusions

The main contribution of this work is to examine global and local versions of the neural models for STLF. The models are analyzed in the context of data representation methods. Daily load curve is introduced to the models as an **x**-vector: the normalized vector of the hourly loads. This preprocessing simplifies the forecasting problem by filtering out both the trend and seasonal variations of periods longer than a day. **X**-vectors express the shape of the daily curve. Similar preprocessing is used for an output variable. The day of the week is encoded in six ways and the hour of the day in five ways. In optimization procedures, the best ways of coding as well as the number of hidden neurons are selected for each forecasting model.

In a global approach, the model is competent for each day of the week and each hour of the day. The relationship between input and output variables is complex in this case, which means a more complex network with more hidden neurons is required. Both the learning and optimization of such model are difficult, time-consuming tasks. While the accuracy of this model is limited, the decomposition of the forecasting problem into subproblems, and modeling these subproblems separately, should lead to an improvement in accuracy. The first decomposition method relies upon splitting the problem into seven subproblems, one for each day of the week individually. However, an experimental study did not confirm better results for this approach. The second decomposition method splits the problem into 24 subproblems, one for each hour of the day. Local NNs are

Table 9 Forecast errors and their interquartile ranges for the proposed and comparative models

Model	PL		FR		GB		VC	
	MAPE	IQR	MAPE	IQR	MAPE	IQR	MAPE	IQR
MLP (v.5)	1.45	1.38	**1.59**	1.64	**1.63**	1.68	**2.99**	2.74
RBFNN	1.67	1.53	1.70	1.70	1.84	1.90	3.23	3.05
GRNN	**1.38**	1.33	**1.64**	1.71	**1.56**	1.64	**2.83**	2.59
FCPNN1	1.71	1.46	1.90	1.95	1.69	1.79	3.18	2.97
FCPNN2	1.63	1.50	1.82	1.86	1.66	1.71	3.22	2.99
SOM1	1.74	1.65	2.10	2.19	1.95	1.98	3.41	3.12
SOM2	1.73	1.53	1.95	2.04	1.78	1.89	3.28	3.08
SOM3	1.99	1.83	2.06	2.18	1.95	2.02	3.63	3.47
PCR	**1.35**	1.33	1.71	1.78	**1.60**	1.68	3.00	2.70
PLSR	**1.34**	1.32	**1.57**	1.61	**1.54**	1.61	**2.83**	2.60
N-WE	**1.30**	1.30	**1.66**	1.67	**1.55**	1.63	**2.82**	2.56
FNM	**1.38**	1.38	**1.67**	1.71	**1.60**	1.66	**2.91**	2.67
FP1 + *k*-means	1.69	1.64	2.05	2.17	1.84	1.88	3.34	3.01
FP2 + *k*-means	1.59	1.51	1.94	2.05	1.76	1.84	3.13	2.94
AIS1	1.50	1.50	1.93	1.95	1.77	1.84	3.04	2.75
AIS2	1.50	1.51	1.93	1.96	1.78	1.87	3.33	2.93
AISLFS	1.51	1.49	1.79	1.81	1.67	1.73	3.13	2.75
ARIMA	1.82	1.71	2.32	2.53	2.02	2.07	3.67	3.42
ES	1.66	1.57	2.10	2.29	1.85	1.84	3.52	3.35
Naïve	3.43	3.42	5.05	5.96	3.52	3.82	4.54	4.20

built for each hour and results are improved. Further improvement is achieved when the problem is decomposed into subproblems representing each day of the week and each hour of the day. In this case, the local relationships between the input and output variables within the subproblems are simpler and can be modeled using less neurons in easy optimization and learning procedures. Finally, the most local decomposition method splits the problem into separate forecasting tasks, i.e., forecasts for a given hour of a given day. In this case, an individual NN learns for each forecasting task and is competent only for this task. New tasks require new NN learning. An advantage of this method is that the model does not require an optimization phase (selection of the number of neurons and data coding method). In its optimal variant, it only has two hidden neurons, so, its learning is very fast. The most local models, v.4 and v.5, reduced the forecast errors significantly, when compared to the global model.

The final recommendation for STLF is using the local MLP models v.4 or v.5 due to the most accurate results. Note that these models have very simple architecture: in v.4 only 2–4 hidden neurons are needed, and in v.5 just two or even one hidden neuron provides sufficiently accurate forecasts. Such simple models learn much faster and are more resistant to overfitting than more complex models v.1, v.2 and v.3. Moreover, the landscape of the error function for them is less complex, so finding a global minimum is more likely.

Compliance with ethical standards

Conflict of interest The author declares that he has no conflict of interest.

References

1. Hippert HS, Pedreira CE, Souza RC (2001) Neural networks for short-term load forecasting: a review and evaluation. IEEE Trans Power Syst 16(1):44–55
2. Kodogiannis VS, Anagnostakis EM (2002) Soft computing based techniques for short-term load forecasting. Fuzzy Sets Syst 128:413–426
3. Dudek G (2016) Neural networks for pattern-based short-term load forecasting: a comparative study. Neurocomputing 2015:64–74

4. Zang H et al (2018) Hybrid method for short-term photovoltaic power forecasting based on deep convolutional neural network. IET Gener Transm Distrib 12(20):4557–4567
5. Kong W, Dong ZY, Jia Y, Hill DJ, Xu Y, Zhang Y (2019) Short-term residential load forecasting based on LSTM recurrent neural network. IEEE Trans Smart Grid 10(1):841–851
6. Wang L, Zhang Z, Chen J (2017) Short-term electricity price forecasting with stacked denoising autoencoders. IEEE Trans Power Syst 32(4):2673–2681
7. Shi H, Xu M, Li R (2018) Deep learning for household load forecasting—a novel pooling deep RNN. IEEE Trans Smart Grid 9(5):5271–5280
8. Rafiei M, Niknam T, Aghaei J, Shafie-Khah M, Catalão JPS (2018) Probabilistic load forecasting using an improved wavelet neural network trained by generalized extreme learning machine. IEEE Trans Smart Grid 9(6):6961–6971
9. Li B, Zhang J, He Y, Wang Y (2017) Short-term load-forecasting method based on wavelet decomposition with second-order gray neural network model combined with ADF test. IEEE Access 5:16324–16331
10. Deihimi A, Orang O, Showkati H (2013) Short-term electric load and temperature forecasting using wavelet echo state networks with neural reconstruction. Energy 57:382–401
11. Kulkarni S, Simon SP, Sundareswaran K (2013) A spiking neural network (SNN) forecast engine for short-term electrical load forecasting. Appl Soft Comput 13(8):3628–3635
12. Papalexopoulos AD, Hao S, Peng T-M (1994) An implementation of a neural network based load forecasting model for the EMS. IEEE Trans Power Syst 9(4):1956–1962
13. Lee KY, Cha YT, Park JH (1992) Short-term load forecasting using an artificial neural network. IEEE Trans Power Syst 7(1):124–132
14. Srinivasan D (1998) Evolving artificial neural networks for short term load forecasting. Neurocomputing 23(1–3):265–276
15. Topalli AK, Erkmen I, Topalli I (2006) Intelligent short-term load forecasting in Turkey. Int J Electr Power Energy Syst 28(7):437–447
16. Methaprayoon K, Lee WJ, Rasmiddatta S, Liao JR, Ross RJ (2007) Multistage artificial neural network short-term load forecasting engine with front-end weather forecast. IEEE Trans Ind Appl 43(6):1410–1416
17. Fan S, Chen L, Lee WJ (2009) Short-term load forecasting using comprehensive combination based on multimeteorological information. IEEE Trans Ind Appl 45(4):1460–1466
18. Cecati C, Kolbusz J, Różycki P, Siano P, Wilamowski BM (2015) A novel RBF training algorithm for short-term electric load forecasting and comparative studies. IEEE Trans Ind Electron 62(10):6519–6529
19. Kalaitzakis K, Stavrakakis GS, Anagnostakis EM (2002) Short-term load forecasting based on artificial neural networks parallel implementation. Electr Power Syst Res 63(3):185–196
20. Kodogiannis VS, Anagnostakis EM (1999) A study of advanced learning algorithms for short-term load forecasting. Eng Appl Artif Intell 12(2):159–173
21. Dillon TS, Sestito S, Leung S (1991) An adaptive neural network approach in load forecasting in a power system. In: Proceedings first international forum on applications of neural networks to power systems, pp 17–21
22. Tamimi M, Egbert R (2000) Short term electric load forecasting via fuzzy neural collaboration. Electr Power Syst Res 56(3):243–248
23. Hanmandlu M, Chauhan BK (2011) Load forecasting using hybrid models. IEEE Trans Power Syst 26(1):20–29
24. Khotanzad A, Hwang RC, Abaye A, Maratukulam D (1995) An adaptive modular artificial neural network hourly load forecaster and its implementation at electric utilities. IEEE Trans Power Syst 10(4):1716–1722
25. Djukanovic M, Ruzic S, Babic B, Sobajic DJ, Pao Y-H (1995) A neural-net based short term load forecasting using moving window procedure. Int J Electr Power Energy Syst 17(6):391–397
26. Hernández L, Baladrón C, Aguiar JM, Carro B, Sánchez-Esguevillas A, Lloret J (2014) Artificial neural networks for short-term load forecasting in microgrids environment. Energy 75:252–264
27. Amjady N, Keynia F (2009) Short-term load forecasting of power systems by combination of wavelet transform and neuro-evolutionary algorithm. Energy 34(1):46–57
28. Chen Y, Luh PB, Guan C, Zhao YG, Michel LD, Coolbeth MA, Friedland PB, Rourke SJ (2010) Short-term load forecasting: similar day-based wavelet neural networks. IEEE Trans Power Syst 25(1):322–330
29. Ding N, Benoit C, Foggia G, Bésanger Y, Wurtz F (2016) Neural network-based model design for short-term load forecast in distribution systems. IEEE Trans Power Syst 31(1):72–81
30. Sun X, Luh PB, Cheung KW, Guan W, Michel LD, Venkata SS, Miller MT (2016) An efficient approach to short-term load forecasting at the distribution level. IEEE Trans Power Syst 31(4):2526–2537
31. Chu WC, Chen YP, Xu ZW, Lee WJ (2011) Multiregion short-term load forecasting in consideration of HI and load/weather diversity. IEEE Trans Ind Appl 47(1):232–237
32. Dudek G (2015) Pattern similarity-based methods for short-term load forecasting—part 1: principles. Appl Soft Comput 37:277–287
33. Ferreira VH, da Silva APA (2007) Toward estimating autonomous neural network-based electric load forecasters. IEEE Trans Power Syst 22(4):1554–1562
34. Dudek G (2013) Forecasting time series with multiple seasonal cycles using neural networks with local learning. In: Rutkowski L et al (eds) Artificial intelligence and soft computing, ICAISC 2013, LNCS 7894, pp 52–63
35. Dudek G (2016) Pattern-based local linear regression models for short-term load forecasting. Electr Power Syst Res 130:139–147
36. Dudek G (2015) Pattern similarity-based methods for short-term load forecasting—part 2: models. Appl Soft Comput 36:422–441
37. Dudek G (2017) Artificial immune system with local feature selection for short-term load forecasting. IEEE Trans Evol Comput 21:116–130